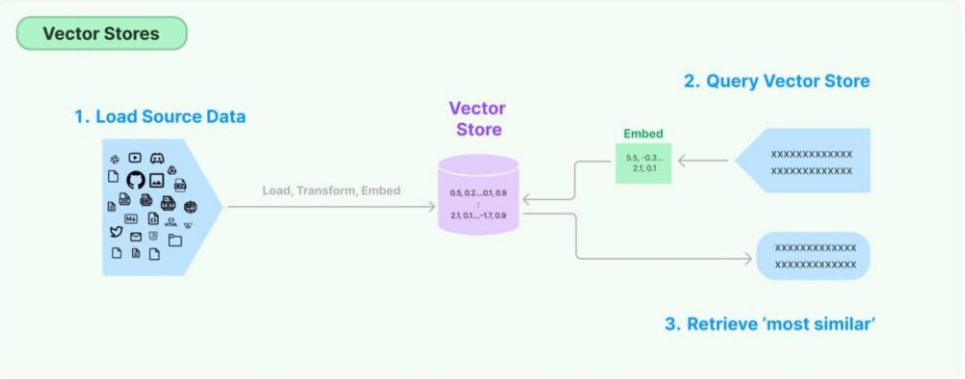
Word Embeddings and Vector Stores

- For an automated search of relevant information, the free-text data needs to be converted into a numerical form that can be compared. That form is embeddings.
- **Embeddings are** high-dimensional vector representations of text, where the position of each vector reflects the semantic meaning of the text it represents. Similar meanings are represented by similar vectors.
- An **Embedding Model** processes each chunk of our knowledge base and converts it into a high-dimensional vector. These vectors are positioned in a semantic vector space such that:
  - **Similar content** is placed **closer together**,
  - **Dissimilar content** is placed **further apart**.
- Once our knowledge base is embedded, the vectors need to be stored in a way that allows for efficient retrieval. That's the job of a **Vector Store**.
- A Vector Store is a database that stores embeddings and supports similarity search using vector distance metrics.

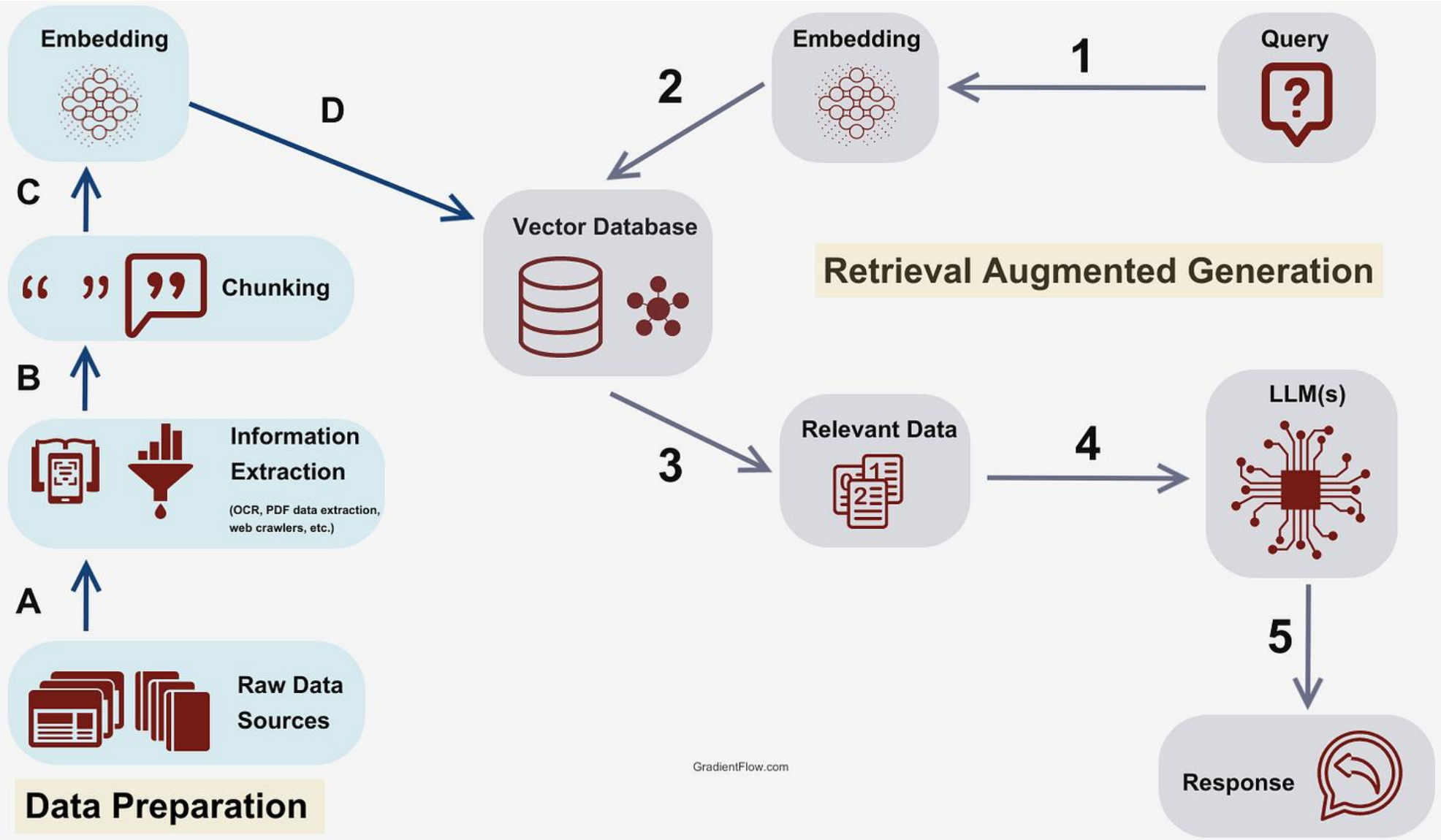
Vector stores

- At its core, a vector essentially is an array of numbers.
- They can be used to represent entities such as words, phrases, images, or audio files within a continuous, high-dimensional space known as an embedding.
- These embeddings effectively map the syntactic and semantic meaning of words or shared features in a wide range of data types.
- They find utility in applications, such as recommendation systems, search algorithms, and even in generating text, akin to the capabilities of ChatGPT.
- Embeddings are stored on special databases called Vector Database.
- Unlike a conventional relational database, which is organized in rows and columns, or a document database with documents and collections, a vector database arranges sets of numbers together based on their similarity.
- This design enables ultra-fast querying, making it an excellent choice for AI-powered applications.
- The surge in popularity of these databases can be attributed to their ability of enhancing and fine-tuning LLMs' capabilities with long-term memory and the possibility to store domain-specific knowledge bases.
- The process involves loading the data sources (be it images, text, audio, etc.) and using an embedder model, for example, OpenAI's Ada-002 or Meta's LLaMA to generate vector representations.
- Next, embedded data is loaded into a vector database, ready to be queried.
- When a user initiates a query, this is automatically embedded and a similarity search across all stored documents is performed.
- In this way, pertinent documents are retrieved from the vector database to augment the context information the model can rely on to generate tailored responses.
- Popular vector store databases are Chroma and FAISS.



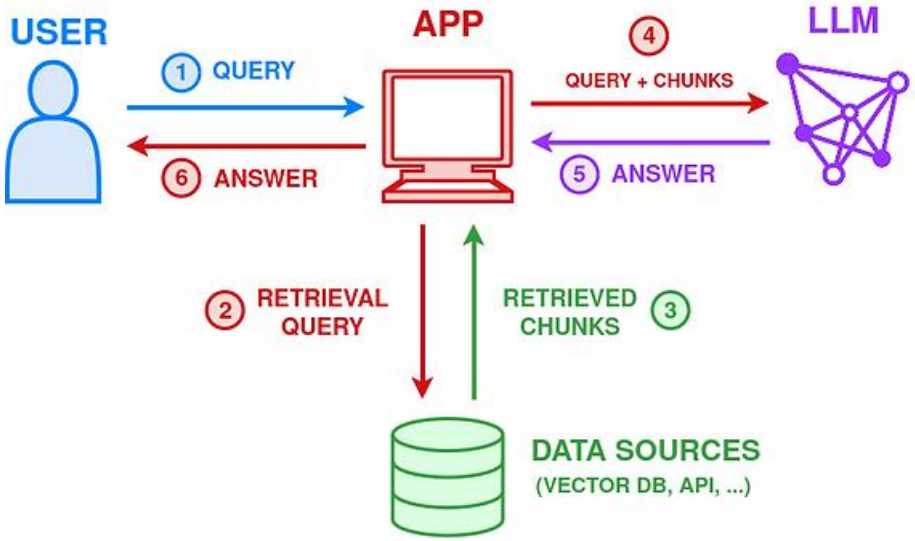
https://www.knime.com/blog/4-levels-llm-customization

General Overview of the RAG Process (Building and Using the Vector Store)



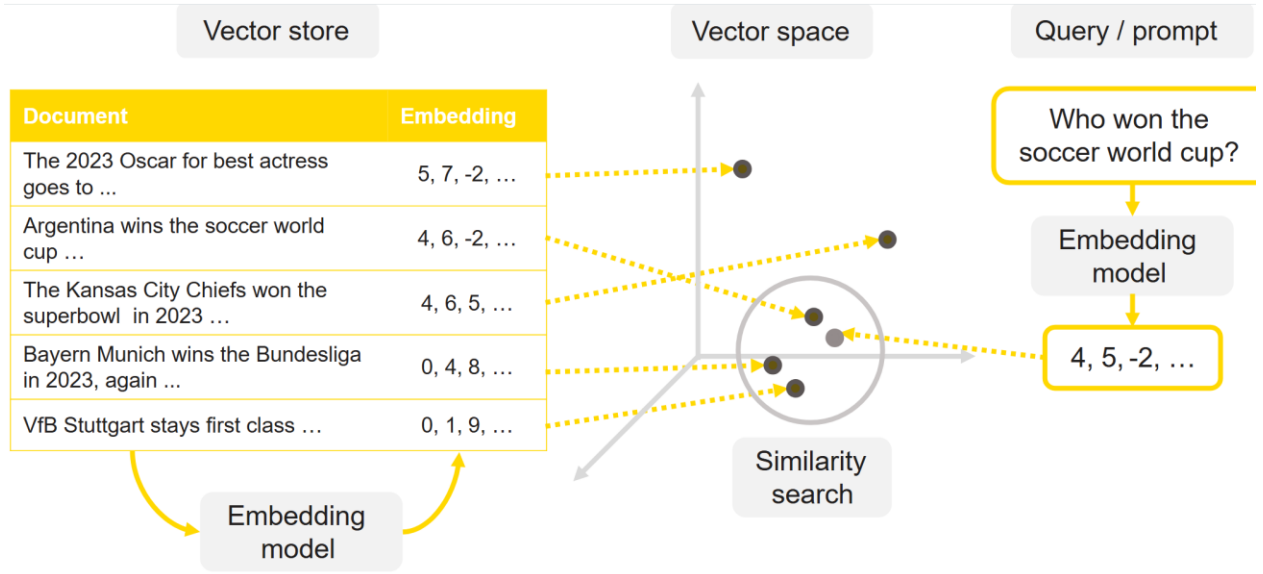
User Query

- So how do embeddings and vector stores enable automated retrieval of the relevant information?
- When a user submits a query:
  - The query is converted into an embedding using the same embedding model.
  - The vector store compares the query embedding to all stored document embeddings.
  - It returns the top k most similar vectors, i.e., the text chunks most relevant to the query.

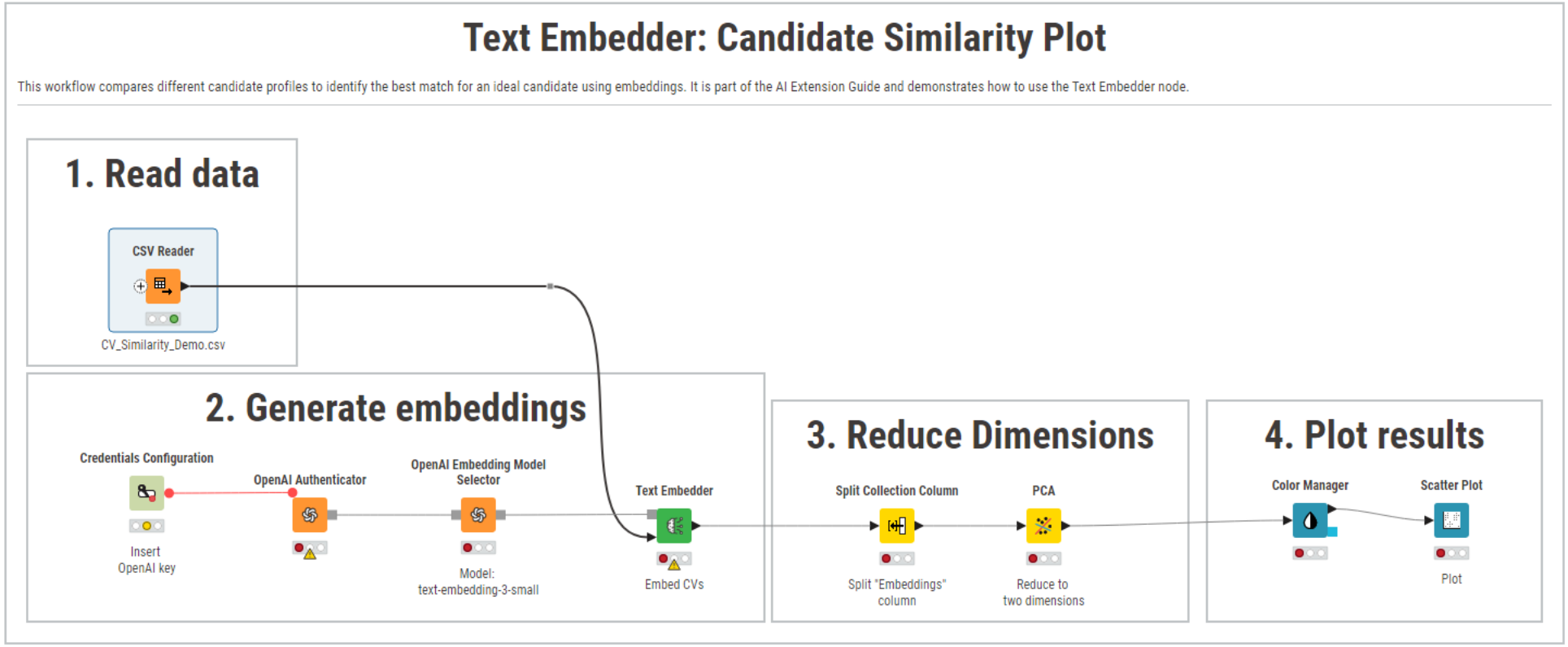


Embeddings and Vector Stores

- In the vector store, a similarity metric determines why the user prompt is more similar to one document over another document.
- The similarity metric computes and assigns a score of similarity to each document to find out how similar these are to the user prompt.
- Next, documents are sorted by their similarity scores from more similar to less similar.
- Different similarity metrics are used depending on the vector store (e.g., Chroma, FAISS, etc.).



Text Embedding Example



This workflow can be downloaded as following:

1. Download Course Workflows from VClass
2. Goto Generative AI Folder -> AI Extension Guide -> 1. Prompting LLM
3. Open Text Embedder